EFFECTIVE MACHINE LEARNING TECHNIQUES TO DETECT FATTY LIVER DISEASE

A

Project Report Submitted

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

Dudekula Nasreen	**19761A0581**
Somu Chowdeswar Reddy	**19761A05B7**
Pajjuru Lasya Sri	**19761A05A8**

Under the esteemed guidance of

Mr. N.V. Naik

Sr. Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Accredited by NAAC with 'A' Grade & NBA (Under Tier - I), ISO 9001:2015 Certified
Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

2019-2023

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)

Accredited by NAAC with 'A' Grade & NBA (Under Tier - I), ISO 9001:2015 Certified
Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

Department of

COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**Effective Machine Learning Techniques To Detect Fatty Liver Disease**" is being submitted by

Dudekula Nasreen	**19761A0581**
Somu Chowdeswar Reddy	**19761A05B7**
Pajjuru Lasya Sri	**19761A05A8**

in partial fulfillment of the requirements for the award of degree of **B.Tech** in **Computer Science & Engineering** from **Jawaharlal Nehru Technological University Kakinada** is a record of bonafide work carried out by them at **Lakireddy Bali Reddy College of Engineering.**
The results embodied in this Project report have not been submitted to any other University or Institute for the award of any degree or diploma.

PROJECT GUIDE
Mr. N.V. Naik

HEAD OF THE DEPARTMENT
Dr. D.Veeraiah

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We take great pleasure to express our deep sense of gratitude to our project guide **Mr. N.V. Naik,** Sr. Assistant Professor, for his valuable guidance during the course of our project work.

We would like to thank **Dr. D. Veeraiah**, Professor & Head of the Department of Computer Science & Engineering for his encouragement.

We would like to express our heart-felt thanks to **Dr. K. Appa Rao,** Principal, Lakireddy Bali Reddy College of Engineering for providing all the facilities for our project.

Our utmost thanks to all the faculty members and Non-Teaching Staff of the Department of Computer Science & Engineering for their support throughout our project work.

Our Family Members and Friends receive our deepest gratitude and love for their support throughout our academic year.

Dudekula Nasreen	**19761A0581**
Somu Chowdeswar Reddy	**19761A05B7**
Pajjuru Lasya Sri	**19761A05A8**

DECLARATION

We are here by declaring that the project entitled **“Effective Machine Learning Techniques To Detect Fatty Liver Disease”** work done by us. We certify that the work contained in the report is original and has been done by us under the guidance of our supervisor. The work has not been submitted to any other institute in preparing for any degree or diploma. We have followed the guidelines provided by the institute in preparing the report. We have confirmed to the norms and guidelines given in the Ethical Code of Conduct of the Institute. Whenever we have used materials (data, theoretical analysis, figures and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references. Further, we have taken permission from the copyright’s owner of the sources, whenever necessary**.**

Signature(s) of the students(s)

Dudekula Nasreen	**19761A0581**
Somu Chowdeswar Reddy	**19761A05B7**
Pajjuru Lasya Sri	**19761A05A8**

ABSTRACT

Heart disease, lung disease, respiratory disease, etc. are currently the top killers. The majority of liver problems are difficult to detect early on. One of these is fatty liver disease, a common disorder brought on by a collection of too much liver fat. Hepatic steatosis is an additional name for it. Drinking heavily makes you more prone to acquire it. Alcoholism causes the cells of the liver to accumulate fat. The liver's ability to function is hampered by this. It could result in cancer of the liver and liver damage. Even if a person does not regularly consume alcohol, they can nonetheless get fatty liver disease. Blood tests, ultrasounds, and computerized tomography scans are the three main types of diagnostic tests. A more precise and dependable, for the early identification of fatty liver disease, automated software is needed. To anticipate the disease, particular machine learning models are created for this purpose. To identify the fatty liver disease with specificity, accuracy, and dependability, methods of Naive Bayes (NB), Random Forest (RF), and Hybrid of ANN with eXtreme Gradient Boosting are proposed in this study. A total of 70000 cases are included in the collection. This classification system is assessed for precision using a confusion matrix.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. XGBoost	Extreme Gradient Boosting
2. ANN	Artificial Neural Network
3. NB	Naïve Bayes
4. RF	Random Forest
5. LR	Linear Regression
6. KNN	K-Nearest Neighbor
7. FLD	Fatty Liver Disease
8. CT	Computed Tomography
9. MRI	Magnetic Resonance Imaging

1. INTRODUCTION

1.1 Overview of The Project

Fatty liver disease (FLD) is a common clinical problem; it is also associated with high morbidity and mortality. FLD eventually leads to noncholestatic cirrhosis and hepatocellular carcinoma. Additionally, FLD has been increasing in parallel with the prevalence of diabetes, metabolic syndrome and obesity. The higher prevalence of FLD has appeared as a greater economic burden. Therefore, accurate identification of individuals at risk and early recognition of FLD could offer immense benefits for diagnosis, preventive or even proper treatment. Over the past decade, biopsy has been used to stratify patients, and considered as a diagnostic reference standard for the assessment of fatty infiltration of the liver. However, this method is highly invasive and costly; it also might trigger side effects and sampling errors during the application of this method. Although, ultrasonography is using as a functional tool for FLD diagnosis with higher accuracy, while identifying accuracy is highly operator dependent.

Machine learning (ML) is a field of computer science that uses computer algorithms to identify patterns in large data, and also assist to predict the various outcomes based on data. ML techniques have emerged as a potential tool for prediction and decision-making in a multitude of disciples. Due to the availability of clinical data, ML has been playing a critical role in medical decision making as well. Developing a machine learning model would serve as a valuable aid to identify disease and make a real-time effective clinical decision. It would also allow for optimization of hospital resources by classifying the right patients with significant several risk factors earlier. Nowadays, many studies have already investigated medical imaging techniques such as ultrasound (US), computed tomography (CT), and magnetic resonance imaging (MRI) for fatty liver disease classification. Ultrasound imaging is noninvasive, inexpensive, easy to operate, and portable.

Andrade et al. evaluated the performance of three classifiers for diagnosis of liver steatosis, using several extracted features from ultrasound images. Ribeiro and Sanches utilized the anatomic and echogenic information of normal liver and fatty liver ultrasonic images, and used Bayesian framework on the extracted feature parameters for fatty liver diagnosis.

Owjimehr et al. demonstrated an automatic ROI selection and hierarchical classification method to discriminate normal and three stages of fatty liver, steatosis, fibrosis, and cirrhosis. Their algorithms discriminated the normal patients from fatty liver patients in the first step by the use of wavelet packet transform (WPT) features, and classified steatosis and the other stages of the fatty liver in the second step by a fusion of WPT and gray-level difference statistical (GLCM) features. Moreover, Li et al. analyzed B-mode ultrasonic images texture features of fatty liver, composed near-field light-spot density, near-far-field grayscale ratio, grayscale co-occurrence matrix, and neighborhood gray-tone difference matrix, and used support vector machine (SVM) as the classification algorithm.

1.2 Feasibility Study

A preliminary investigation examines project feasibility, and the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Social and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

1.2.1 Economical Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.
The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economic feasibility for certain.

1.2.2 Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

• Does the necessary technology exist to do what is suggested?
• Do the proposed equipment have the technical capacity to hold the data required to use the new system?
• Will the proposed system provide an adequate response to inquiries, regardless of the number or location of users?
• Can the system be upgraded if developed?
• Are there technical guarantees of accuracy, reliability, ease of access, and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus it provides easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides a technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology.

1.2.3 Social Feasibility

The use of machine learning algorithms such as XGBoost (eXtreme Gradient Boosting) and ANN (Artificial Neural Networks) for detecting fatty liver disease is a promising approach, and has the potential to provide an accurate and efficient diagnosis of this condition.

From a social feasibility perspective, there are several factors that could impact the adoption and implementation of such a system:

Accessibility: The availability and accessibility of these technologies could be a challenge in certain regions or communities where access to healthcare is limited.

Cost: The cost of implementing such a system may be a barrier for some healthcare providers, especially in developing countries or underfunded healthcare systems.

Ethical concerns: There may be concerns related to the ethical implications of using machine learning for medical diagnosis, including issues of privacy, informed consent, and bias.

Training and education: Healthcare professionals may require additional training and education to effectively utilize and interpret the results of machine learning algorithms.

Overall, the use of machine learning for detecting fatty liver disease has the potential to improve diagnosis and treatment outcomes. However, it is important to consider the social and ethical implications of such systems, and ensure that they are accessible, affordable, and provide accurate and unbiased results.

1.3 Scope

Fatty liver disease is a condition in which there is an excessive accumulation of fat in the liver. This can lead to inflammation, scarring, and in severe cases, liver failure. Detecting fatty liver disease early is important as it can help prevent further damage to the liver and improve patient outcomes. Machine learning techniques can be used to analyze medical images and other clinical data to detect fatty liver disease with high accuracy. The scope of effective machine learning techniques to detect fatty liver disease involves the following:

Data collection: The first step in any machine learning project is to collect relevant data. In the case of fatty liver disease, this includes medical images such as ultrasound, computed tomography (CT), and magnetic resonance imaging (MRI), as well as clinical data such as patient demographics, laboratory results, and medical history.

Data preprocessing: Once the data is collected, it needs to be preprocessed to ensure that it is clean and ready for analysis. This involves tasks such as data cleaning, feature engineering, and normalization.

Machine learning algorithm selection: There are a variety of machine learning algorithms that can be used to detect fatty liver disease. These include logistic regression, random forests, support vector machines (SVM), and deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Model training and validation: Once the machine learning algorithm is selected, the model needs to be trained on the data. This involves splitting the data into training and validation sets and using the training data to optimize the model's parameters. The validation set is then used to evaluate the model's performance and make any necessary adjustments.

Model deployment: After the model has been trained and validated, it can be deployed in a clinical setting. This involves integrating the model into clinical workflows and ensuring that it is accurate and reliable.

Overall, the scope behind effective machine learning techniques to detect fatty liver disease involves a combination of data collection, preprocessing, algorithm selection, model training and validation, and model deployment. By using these techniques, it is possible to develop accurate and reliable models for detecting fatty liver disease, which can help improve patient outcomes and reduce the burden of liver disease on healthcare systems.

2.LITERATURE SURVEY

Machine Learning Algorithms for Predicting Fatty Liver Disease

Pei X et al proposed a model to predict FLD that can support medical professionals in catogorising people who are at high risk of FLD and in making unique diagnoses, decisions about treatment, and plans for FLD prevention. A total of 3,419 participants were chosen, and 845 of them had FLD screenings. In order to find the disease, classification models were applied. The models included in this study are LDA, KNN, ANN, LR, RF and XGBoost. The prediction accuracy was measured using AUC, sensitivity, specificity, positive predictive value, and negative predictive value. It demonstrated that machine learning models yield more accurate predictions, with the XGBoost model having the best accuracy.

A Machine Learning Based Framework to Identify and Classify Non-alcoholic Fatty Liver Disease in a Large-Scale Population

Weidong Ji et al created and evaluated machine learning (ML) models that can be utilised for identifying a group of individuals. This research involved 304,145 individuals who took part in the national physical examination, and their survey results and physical measurement data were used as candidate covariates in the model. The relevance score of the covariate in NAFLD after absolute shrinkage was generated by a classifier with the highest performance, and a selection operator (LASSO) was used to feature select from potential covariates. The screening model for NAFLD was then developed using four ML approaches. The performance of XGBoost was the best of the four ML algorithms, with BMI, age and waist circumference ranking highest in significance.

Prediction of fatty liver disease using machine learning algorithms

Chieh-ChenWu et al. set out to design a model to predict FLD that would help medical practitioners categories a patients, establish a diagnosis, and treat, and stop FLD. To predict FLD, classification algorithms such as RF, NB, ANN, and LR were developed. The area under the receiver operating characteristic curve was used to assess the performance of the four models (ROC). The experiment involved 577 individuals, 377 of whom had fatty livers. The random forest model outperformed the others.

Application of Machine Learning Techniques for Clinical Predictive Modeling: A Cross-Sectional Study on Nonalcoholic Fatty Liver Disease in China

Cheng-fu Xu et al proposed the best clinical prediction model for NAFLD was assessed using machine learning techniques. At Zhejiang University, participants in a health examination participated in a cross-sectional study. The use of questionnaires, lab testing, physical exams, and hepatic ultrasonography was made. Then, using the free program Weka, machine learning techniques were put into practice. Features selection and classification were among the tasks. A screening model was created using feature selection approaches by deleting unnecessary elements. A prediction model was created using classification and assessed using the F-measure. 11 cutting-edge machine learning methods were researched. 2,522 (24%) of the 10,508 registered participants matched the NAFLD diagnostic criteria. Using a variety of statistical testing methodologies, the top five risk variables for NAFLD were discovered to be BMI, triglycerides, gamma-glutamyl transpeptidase (GT), serum alanine aminotransferase (ALT), and uric acid. To classify the data, a 10-fold cross-validation was used. The results revealed that, among the 11 different tactics tested, the Bayesian network model performed the best. For accuracy, specificity, sensitivity, and F-measure, up to 83%, 0.878, 0.675, and 0.655, respectively, were obtained. The Bayesian network model increases the F-measure score by 9.17% when compared to logistic regression.

2.1 Existing System & Drawbacks

Naive Bayes (NB) is a generative model that makes dealing with missing values a lot easier. It is a classification model which predicts a class label y given a feature vector x = [x1, x2, x3…xd] T and helps to make an inference on a new sample xnew = [x1, x2, x3…xd] T with a missing feature xm. It is yet very powerful model that is used to return not the prediction but also the degree of certainty. It is very easy to understand and implement.

Drawbacks:

- Lack of accuracy.
- More number of iterations.
- To detect FLD is a difficult task and it's detecting at the last stages.
- By using the number of filters we are detecting the FLD which increases the cost of the system and also expert person needs to be there to detect the FLD.

2.2 Proposed System

In the field of computer science, data mining, indicating the extraction of focused information from a larger data set, is a modern term describing the approach for analyzing big data sets. Machine learning (ML) algorithms are data-mining tools. Machine learning refers to a variety of techniques dealing with pattern recognition based on models for classification and the prediction of new data. ANN(Artificial Neural Network) and XG-BOOST.

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient-boosting framework. In prediction problems involving unstructured data (images, text, etc.) XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Applying ANN to the data set alone gives fewer performance metrics 70 percent accuracy is found which is not a good model prediction. And another XG-Boost algorithm also alone when applied gives the result of performance 82 accuracy which can be improved. Hence by using a hybrid of these algorithms we can evaluate the better performance of our model and obtain higher accuracy comparatively.

Advantages:

- This is faster when compared with other algorithms.
- This shows more accuracy in finding the disease prediction.
- Very less number of iterations are used by this algorithm.

3.SYSTEM ANALYSIS

3.1 Overview of System Analysis

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems.

Software Model or Architecture Analysis:

In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system in the software development process.

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management.

In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three-dimensional model which comprises the user, owner and developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three-dimensional model which comprises of the user, owner and the developer.

One size fits all approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above-mentioned defects by using a new hypothetical model for SDLC described elsewhere.

The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

What is SDLC?

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC). A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.

SDLC models * The Linear model (Waterfall) - Separate and distinct phases of specification and development. - All activities in linear fashion. - Next phase starts only when first one is complete. Evolutionary development - Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development). - Incremental Model (Waterfall in iteration), - RAD(Rapid Application Development) - Focus is on developing quality product in less time, - Spiral Model - We start from smaller module and keeps on building it like a spiral. It is also called Component based development. * Formal systems development - A mathematical system model is formally transformed to an implementation. * Agile Methods. - Inducing flexibility into development. * Reuse-based development - The system is assembled from existing components.

3.1.1 Requisites Accumulating and Analysis

Accumulating and analyzing data is a crucial step in developing effective machine learning techniques for detecting fatty liver disease. Here are some of the key requisites for this process:

Access to high-quality data: To develop accurate and reliable machine learning models for detecting fatty liver disease, you need access to high-quality data. This includes medical images such as ultrasound, CT, and MRI scans, as well as clinical data such as patient demographics, laboratory results, and medical history. The data should be representative of the patient population you are trying to diagnose, and it should be well-curated and annotated to ensure that it is accurate and reliable.

Data preprocessing: Once you have collected your data, you need to preprocess it to prepare it for analysis. This involves tasks such as data cleaning, feature engineering, and normalization. Data cleaning involves removing any data points that are incomplete, inconsistent, or irrelevant to the analysis. Feature engineering involves selecting the most important features or variables to include in the analysis, and normalization involves scaling the data to ensure that it is on a consistent scale.

Algorithm selection: There are a variety of machine learning algorithms that can be used to detect fatty liver disease, including logistic regression, random forests, SVMs, and deep learning models such as CNNs and RNNs. The choice of algorithm will depend on factors such as the size and complexity of the data, the desired accuracy and speed of the model, and the available computational resources.

Model training and validation: Once you have selected your algorithm, you need to train and validate your model on the data. This involves splitting the data into training and validation sets, using the training data to optimize the model's parameters, and then evaluating the model's performance on the validation set. This process may need to be repeated multiple times with different algorithms or parameters to find the best-performing model.

Deployment and monitoring: Once you have developed an effective machine learning model for detecting fatty liver disease, you need to deploy it in a clinical setting. This involves integrating the model into clinical workflows and ensuring that it is accurate and reliable. You also need to monitor the model's performance over time to ensure that it remains effective as new data becomes available.

Overall, accumulating and analyzing data is a complex and iterative process that requires careful attention to detail and specialized expertise. By following best practices for data collection, preprocessing, algorithm selection, and model training and validation, you can develop accurate and reliable machine learning techniques for detecting fatty liver disease that can improve patient outcomes and reduce the burden of liver disease on healthcare systems.

3.1.2 System Design

Designing a system for detecting fatty liver disease using machine learning techniques involves several key components:

Data collection and preprocessing: The system needs to collect medical images and clinical data from patients, preprocess it to ensure it is of high quality and in a format suitable for analysis.

Feature extraction: The system should extract relevant features from the medical images and clinical data, such as liver size, texture, and density, as well as patient demographics, laboratory results, and medical history.

Machine learning algorithm selection: The system should choose the most appropriate machine learning algorithm to use, based on the type and amount of data available, the desired accuracy and speed of the model, and the computational resources available.

Model training and validation: The system needs to train and validate the machine learning model on the collected data, and test it on new data to ensure it generalizes well to unseen data. This process may need to be repeated multiple times with different algorithms or parameters to find the best-performing model.

Integration and deployment: Once the model is developed, the system should integrate it into the clinical workflow, such as a radiology department or a hospital electronic health record system, and make it accessible to clinicians. The system should also monitor the model's performance over time, and update it as needed to ensure continued accuracy and reliability.

Ethical and regulatory considerations: The system should also take into account ethical and regulatory considerations, such as patient privacy and informed consent, and adhere to relevant data protection regulations and guidelines.

Designing an effective system for detecting fatty liver disease requires careful consideration of these key components, as well as specialized expertise in machine learning, data science, and software engineering. However, a well-designed system can improve patient outcomes by enabling earlier detection and intervention for liver disease, as well as reducing the burden on healthcare systems.

3.1.3 Implementation

Implementing a system for detecting fatty liver disease using machine learning techniques requires specialized expertise in machine learning, data science, and software engineering. However, a well-implemented system can improve patient outcomes by enabling earlier detection and intervention for liver disease, as well as reducing the burden on healthcare systems.

3.1.4 Testing

Testing is a critical step in the development of a system for detecting fatty liver disease using machine learning techniques, and should be performed thoroughly and rigorously to ensure its accuracy, reliability, and effectiveness in clinical practice.

3.1.5 Deployment of System

Deployment of a system for detecting fatty liver disease using machine learning techniques requires careful planning and execution to ensure its effectiveness, safety, and usability in clinical practice. By following best practices in system deployment and maintenance, healthcare providers can leverage the power of machine learning to improve patient outcomes and reduce the burden on healthcare systems.

3.1.6 Maintenance

Maintenance of a system for detecting fatty liver disease using machine learning techniques is a critical aspect of its ongoing success in clinical practice. By monitoring and addressing issues regularly, healthcare providers can ensure that the system remains accurate, reliable, and effective in improving patient outcomes and reducing healthcare costs.

3.2 Software Used in The Project

Pandas

Pandas is an open-source, Python library providing high-performance, easy-to- use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Matplotlib

Matplotlib is probably the single most used Python package for 2D-graphics. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats. We are going to explore matplotlib in interactive mode covering most common cases.

Sklearn

scikit-learn, commonly referred to as sklearn, is a popular open-source machine learning library for Python. It provides a range of tools for data preprocessing, feature selection, model selection, and evaluation, as well as a variety of machine learning algorithms, such as decision trees, random forests, support vector machines, and neural networks.

Some of the key features of scikit-learn include:

Consistent API: scikit-learn provides a consistent API across all its tools and algorithms, which makes it easy to use and learn.

Extensive documentation: scikit-learn has extensive documentation that includes tutorials, user guides, and API references, making it easy to get started and find help.

Data preprocessing: scikit-learn provides a range of tools for data preprocessing, including data cleaning, feature scaling, and feature extraction.

Model selection: scikit-learn provides tools for model selection, such as cross-validation and grid search, which help in selecting the best model for a given problem.

Evaluation metrics: scikit-learn provides a range of evaluation metrics, such as accuracy, precision, recall, and F1-score, which help in evaluating the performance of machine learning models.

Integration with other libraries: scikit-learn can be easily integrated with other Python libraries, such as NumPy, Pandas, and Matplotlib, which makes it a powerful tool for data analysis and visualization.

Overall, scikit-learn is a powerful and user-friendly machine learning library that is widely used in both academia and industry for a variety of applications, including medical image analysis, natural language processing, and computer vision.

3.3 SYSTEM REQUIREMENTS

3.3.1 Software Requirements

- OS: Windows or Linux
- Python IDE : python 2.7.x and above
- Pycharm IDE Required, jupyter notebook
- Setup tools and pip to be installed for 3.6 and above
- Language : Python Scripting

3.3.2 Hardware Requirements

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk: 500GB: Minimum

4. SYSTEM DESIGN

4.1 Overview of System Design

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. The design document is similar to a blue print or plan for the solution, and is used for later during implementation, testing and maintenance.

The design activity is often divided into two separate phase-system design and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided.

A design methodology is a systematic approach to creating a design by application f set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it is partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for properly designing the part. For this, abstraction is used.

System Design:

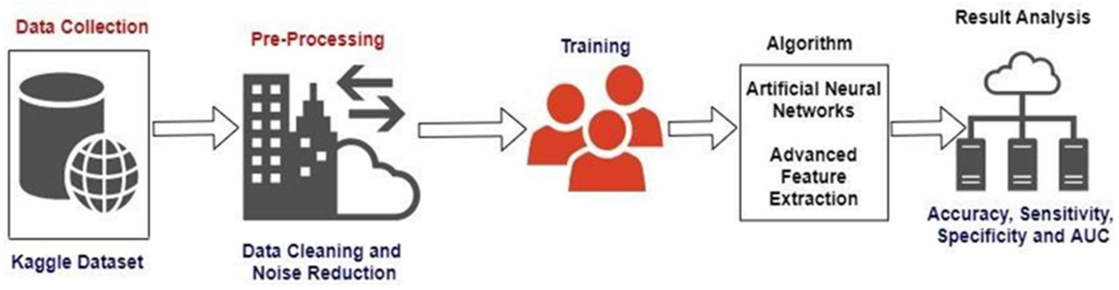

Data Collection
Kaggle Dataset
Pre-Processing
Data Cleaning and Noise Reduction
Training
Algorithm
Artificial Neural Networks
Advanced Feature Extraction
Result Analysis
Accuracy, Sensitivity, Specificity and AUC

Fig. 1. Design of Fatty Liver Detection

The design of the project is described as , the user has the predefined data set which contains all the information related to credit card holders. Later, Python libraries are imported for the data set. The libraries include NumPy, Pandas, Sklearn, Matpotlib.

Pandas is an open-source, Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding. Matplotlib is probably the single most used Python package for 2D- graphics. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats. We are going to explore matplotlib in interactive mode covering most common cases.

Then we use machine learning algorithms such as NB, RF, Hybrid of ANN with XGBoost. Here we use logistic regression. The algorithm applies on the data and the result will be in the form of plotting.

INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- ➢ What data should be given as input?
- ➢ How the data should be arranged or coded?
- ➢ The dialog to guide the operating personnel in providing input.
- ➢ Methods for preparing input validations and steps to follow when error occur.

Objectives

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

Input Design

1. In this system load the fatty liver dataset.

2. The preprocessing and training is done in this step.

3. Applying the proposed algorithm.

Output design

1. In this place user can login after that he will migrate vm's from one (A) host to another host (B).

2. He will measure downtime and migration time.

5.CODING & IMPLEMENTATION

In this phase the design is translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high-level programming languages like C, C++, Pascal, Java, .Net are used for coding. With respect to the type of application, the right programming language is chosen.

Python:

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

CODING (SOURCE CODE)

We load a dataset using Pandas library, and apply the following algorithms and the find the best one for the specific dataset by accuracy evaluation methods

ui.py: This is the user interface file which consists of events for all the algorithms and this is starting page of the project.

```
from tkinter import *
from tkinter.ttk import *
import os
```

```
# Top level window
root = Tk()

root.title("Prediction Of Fatty Liver Disease Using Machine Learning Algorithms")
root.geometry('600x300')

var =  IntVar()

def sel():
    print('')

style = Style(root)
style.configure("TRadiobutton", background = "light green",
                foreground = "red", font = ("arial", 10, "bold"))

ans1 = Radiobutton(root, text='NB-Existing System', variable=var, value=1,command=sel)
ans2 = Radiobutton(root, text='RF-Proposed System', variable=var, value=2,command=sel)
ans3 = Radiobutton(root, text='XGB-Enhanced System', variable=var, value=3,command=sel)
ans4 = Radiobutton(root, text='Quit', variable=var, value=4,command=sel)

ans1.pack()
ans2.pack()
ans3.pack()
ans4.pack()

def out():
    global ans1, ans3, ans3 ,ans4
    answer = (ans1 or ans2 or ans3 or ans4(var.get()))
    system = (var.get())
    if system==1:
        os.system('python es.py')
    if system==2:
        os.system('python ps.py')
    if system==3:
        os.system('python enh.py')
    if system==4:
        root.destroy()

button = Button(root,text = "Proceed",command = out)
button.pack()
root.mainloop()
```

Existing System (es.py) file

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from tkinter import *
from tkinter import messagebox
import os

hd = pd.read_csv("flp_ds.csv", sep = ",")

print(hd.shape)
print(hd.info())
print(hd.head())

total = hd.isnull().sum().sort_values(ascending =False)
percent = (hd.isnull().sum()/hd.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total,percent], axis = 1, keys = ['Total', 'Percent'])

# Split train and test data
x = hd.drop("flp", axis=1)
y = hd["flp"]

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =0.2, random_state=42)
```

```
#print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

print('\nTraining Size : ')
print(y_train.shape)
print('\nTesting Size : ')
print(y_test.shape)

# Naive bayes
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(x_train, y_train)
Y_pred = gaussian.predict(x_test)
acc_gaussian = round(gaussian.score(x_test, y_test) * 100, 2)

os.system('Keras 118 e '+str(acc_gaussian))

rez = ''
# Using readlines()
file1 = open('output/estats.txt', 'r')
Lines = file1.readlines()

count = 0
# Strips the newline character
for line in Lines:
    count += 1
    #print("Line{}: {}".format(count, line.strip()))
    rez = rez + line.strip()+'\n';

root = Tk()
root.geometry("300x200")
w = Label(root, text =rez, font = "50")
w.pack()
root.mainloop()
```

Random forest classifier (ps.py)

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from tkinter import *
from tkinter import messagebox
import os

hd = pd.read_csv("flp_ds.csv", sep = ",")

print(hd.shape)#shows no.of rows and columns
print(hd.info())#description of the dataset
print(hd.head())#shows the first 5 rows by default

total = hd.isnull().sum().sort_values(ascending =False)
percent = (hd.isnull().sum()/hd.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total,percent], axis = 1, keys = ['Total', 'Percent'])

print(missing_data)

x = hd.drop("flp", axis=1)#input features
y = hd["flp"]#output class

# Split train and test data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =0.2, random_state=42)

#print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

print('\nTraining Size : ')
print(y_train.shape)
print('\nTesting Size : ')
print(y_test.shape)
```

```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(x_train, y_train)
Y_pred = random_forest.predict(x_test)
random_forest.score(x_test, y_test)
acc_random_forest = round(random_forest.score(x_test, y_test) * 100, 2)

print(acc_random_forest)

os.system('Keras 118 p '+str(acc_random_forest))

rez = ''
# Using readlines()
file1 = open('output/pstats.txt', 'r')
Lines = file1.readlines()
count = 0

# Strips the newline character

for line in Lines:
    count += 1
    #print("Line{}: {}".format(count, line.strip()))
    rez = rez + line.strip()+'\n';

root = Tk()
root.geometry("300x200")
w = Label(root, text =rez, font = "50")
w.pack()
root.mainloop()
```

Enhancement (XGBoost and ANN) (En.py)

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
import numpy as np
import itertools
from tkinter import *
from tkinter import messagebox
import os

my_listLabels = []
my_list = []

hd = pd.read_csv("flp_ds.csv", sep = ",")

print(hd.shape)
print(hd.info())
print(hd.head())

total = hd.isnull().sum().sort_values(ascending =False)
percent = (hd.isnull().sum()/hd.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total,percent], axis = 1, keys = ['Total', 'Percent'])

# Split train and test data
x = hd.drop("flp", axis=1)
y = hd["flp"]

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =0.2, random_state=42)

#print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
print('\nTraining Size : ')
print(y_train.shape)
print('\nTesting Size : ')
print(y_test.shape)

# Xgboost Classifier
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
xg = XGBClassifier()
xg.fit(x_train, y_train)
y_pred = xg.predict(x_test)
acc_xg = round(xg.score(x_test, y_test) * 100, 2)

os.system('Keras 118 ann '+str(acc_xg))

rez = ''
# Using readlines()
file1 = open('output/xstats.txt', 'r')
Lines = file1.readlines()

count = 0
# Strips the newline character
for line in Lines:
    count += 1
    #print("Line{}: {}".format(count, line.strip()))
    rez = rez + line.strip()+'\n';

root = Tk()
root.geometry("300x200")
w = Label(root, text =rez, font = "50")
w.pack()
root.mainloop()
```

```
p1 = []
p2 = []
p3 = []
p4 = []
p5 = []

with open("output/estats.txt", encoding="utf-8") as f:
    cnt = 0
    for line in f:
        cnt = cnt+1
        result = [line.strip() for line in line.split(':')]
        if cnt == 1:
            p1.append(float(result[1]))
        if cnt == 2:
            p2.append(float(result[1]))
        if cnt == 3:
            p3.append(float(result[1]))
        if cnt == 4:
            p4.append(float(result[1]))
        if cnt == 5:
            p5.append(float(result[1]))

with open("output/pstats.txt", encoding="utf-8") as f:
    cnt = 0
    for line in f:
        cnt = cnt+1
        result = [line.strip() for line in line.split(':')]
        if cnt == 1:
            p1.append(float(result[1]))
        if cnt == 2:
            p2.append(float(result[1]))
        if cnt == 3:
            p3.append(float(result[1]))
        if cnt == 4:
            p4.append(float(result[1]))
        if cnt == 5:
            p5.append(float(result[1]))

with open("output/xstats.txt", encoding="utf-8") as f:
    cnt = 0
    for line in f:
        cnt = cnt+1
        result = [line.strip() for line in line.split(':')]
        if cnt == 1:
            p1.append(float(result[1]))
        if cnt == 2:
            p2.append(float(result[1]))
        if cnt == 3:
            p3.append(float(result[1]))
        if cnt == 4:
            p4.append(float(result[1]))
        if cnt == 5:
            p5.append(float(result[1]))

###########################

my_listLabels = []
my_list = []

my_listLabels.append('NB')
my_list.append(p1[0])
my_listLabels.append('RF')
my_list.append(p1[1])
my_listLabels.append('Hyb(ANN+XGB)')
my_list.append(p1[2])
```

```
# Plot the bar graph
plot = plt.bar(my_listLabels,my_list)

# Add the data value on head of the bar
for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,1.002*height,'%f' % float(height), ha='center',
va='bottom')

# Add labels and title
plt.title("Fatty Liver Prediction Accuracy")
plt.xlabel("Classifier")
plt.ylabel("Score")

# Display the graph on the screen
plt.show()

##############################

############################

my_listLabels = []
my_list = []

my_listLabels.append('NB')
my_list.append(p2[0])
my_listLabels.append('RF')
my_list.append(p2[1])
my_listLabels.append('Hyb(ANN+XGB)')
my_list.append(p2[2])

# Plot the bar graph
```

```
plot = plt.bar(my_listLabels,my_list)

# Add the data value on head of the bar
for value in plot:
   height = value.get_height()
   plt.text(value.get_x() + value.get_width()/2.,1.002*height,'%f' % float(height), ha='center',
va='bottom')

# Add labels and title
plt.title("Fatty Liver Prediction Sensivity")
plt.xlabel("Classifier")
plt.ylabel("Score")

# Display the graph on the screen
plt.show()

##############################

############################

my_listLabels = []
my_list = []

my_listLabels.append('NB')
my_list.append(p3[0])
my_listLabels.append('RF')
my_list.append(p3[1])
my_listLabels.append('Hyb(ANN+XGB)')
my_list.append(p3[2])

# Plot the bar graph
plot = plt.bar(my_listLabels,my_list)
```

```
# Add the data value on head of the bar
for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,1.002*height,'%f' % float(height), ha='center',
va='bottom')

# Add labels and title
plt.title("Fatty Liver Prediction Specificity")
plt.xlabel("Classifier")
plt.ylabel("Score")

# Display the graph on the screen
plt.show()

##############################

############################

my_listLabels = []
my_list = []

my_listLabels.append('NB')
my_list.append(p4[0])
my_listLabels.append('RF')
my_list.append(p4[1])
my_listLabels.append('Hyb(ANN+XGB)')
my_list.append(p4[2])

# Plot the bar graph
plot = plt.bar(my_listLabels,my_list)

# Add the data value on head of the bar
for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,1.002*height,'%f' % float(height), ha='center',
va='bottom')

# Add labels and title
plt.title("Fatty Liver Prediction AUROC")
plt.xlabel("Classifier")
plt.ylabel("Score")

# Display the graph on the screen
plt.show()

##############################
```

6.SYSTEM TESTING

6.1 Overview of Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 Types of Tests

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input: identified classes of valid input must be accepted. Invalid Input: identified classes of invalid input must be rejected. Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised. Systems/Procedure: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.2.5 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways one is on screen and another in printed format.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

System Testing

Testing Methodologies

The following are the Testing Methodologies:

- ➢ Unit Testing
- ➢ Integration Testing
- ➢ User Acceptance Testing
- ➢ Output Testing
- ➢ Validation Testing

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing. During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

A.Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.
In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

B.Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

• The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

• A driver (i.e.) the control program for testing is written to coordinate test case input and output.

• The cluster is tested.

• Drivers are removed and clusters are combined moving upward in the program Structure

The bottom up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

• Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

• Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

User Training

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

Maintenance

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing are simple and easy to understand which will make maintenance easier.

Testing Strategy

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co- operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

S.NO	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1.	Input the CSV File	Predict the fatty liver in given samples	Fatty liver detected in given dataset	PASS
2.	Result for Accuracy	Showing the Accuracy	Accuracy of Proposed System shown	PASS
3.	Result for sensitivity	Showing the sensitivity	Sensitivity of proposed System shown	PASS
4.	Result for specificity	Showing the specificity	Specificity of proposed System shown	PASS

Table 1: Testing Strategy

7.RESULTS

	Naïve Bayes	Random Forest	Hybrid (ANN+XGBoost)
Accuracy	62.570455	82.424546	86.8836
Sensitivity	62.676507	82.530598	86.9896
Specificity	62.348799	82.202890	86.6619
AUROC	0.788671	0.888671	0.9060

Table 2: Comparison Table

Accuracy of overall fatty liver predicted in given samples

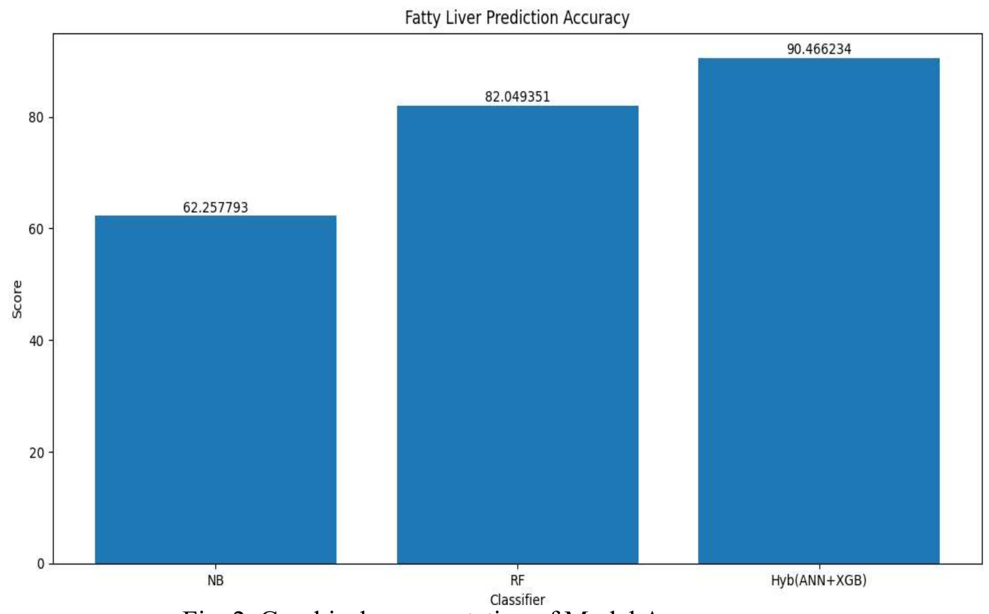


Fig. 2. Graphical representation of Model Accuracy

Sensitivity of overall fatty liver predicted in given samples

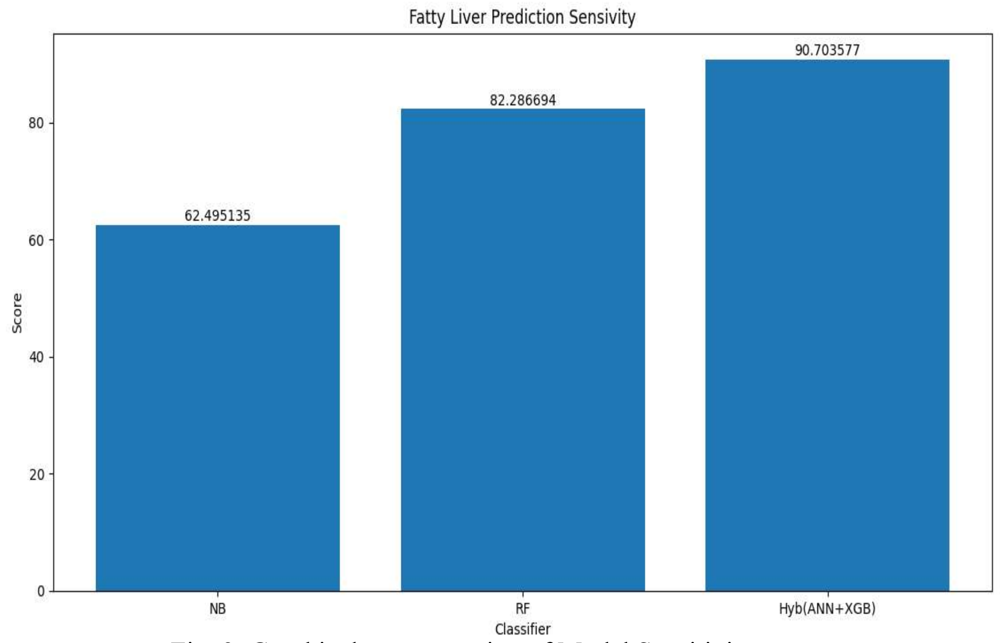


Fig. 3. Graphical representation of Model Sensitivity

Specificity of overall fatty liver predicted in given samples

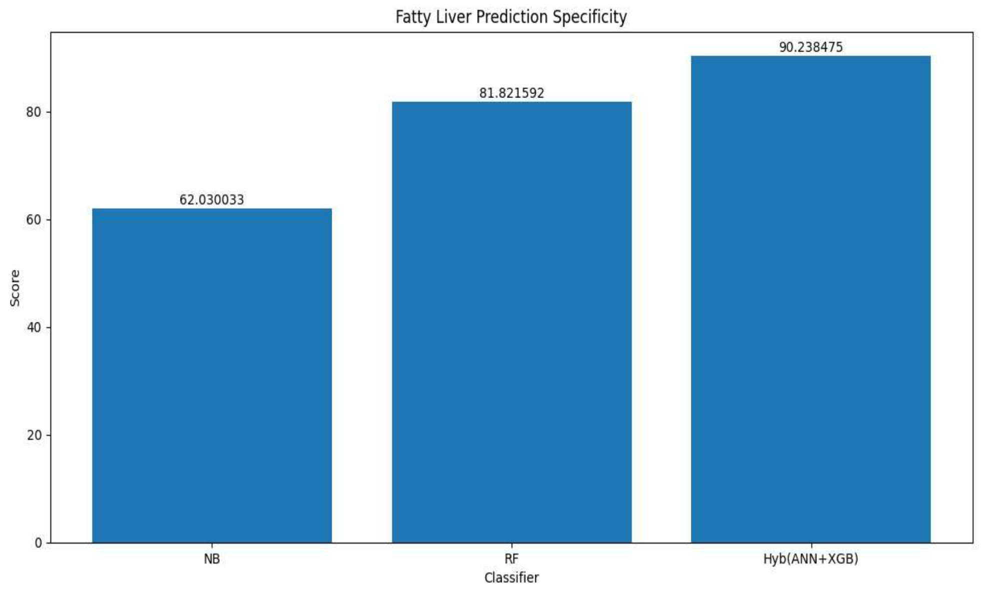


Fig. 4. Graphical representation of Model Specificity

AUROC of overall fatty liver predicted in given samples

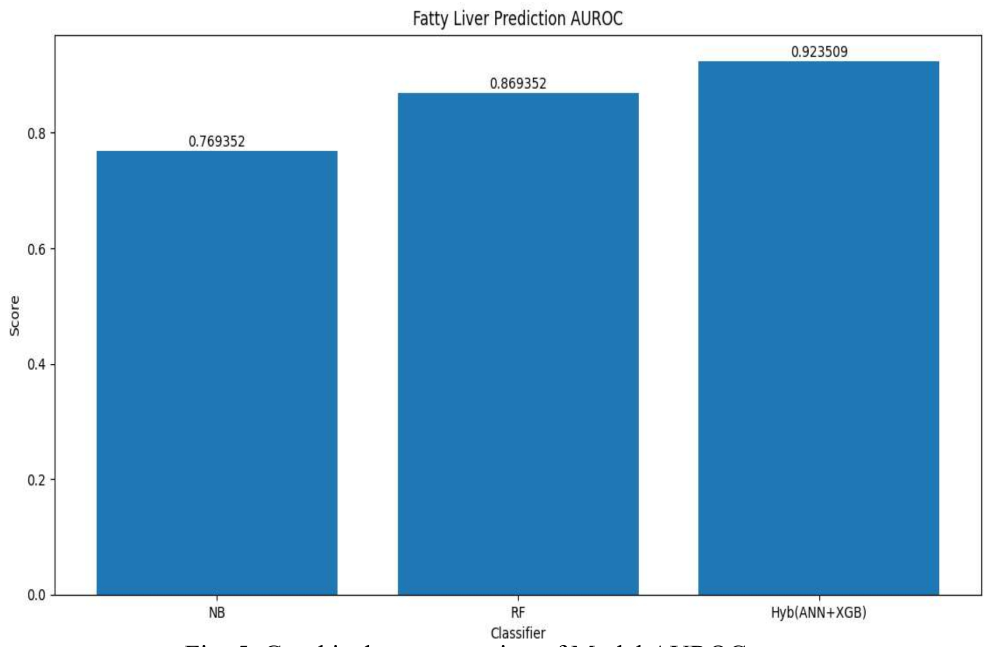


Fig. 5. Graphical representation of Model AUROC

8.CONCLUSION

The three machine learning methods used in this work are contrasted in order to accurately predict fatty liver disease. The hybrid of ANN with XGBoost model, however, demonstrated. superior performance than conventional machine learning methods. Implementing a hybrid ANN-XGBoost approach in the clinical setting could assist doctors in classifying individuals with monitoring, early treatment, and care for fatty liver.

9.REFERENCES

[1] Pei X, Deng Q, Liu Z, Yan X, Sun W: Machine Learning Algorithms for Predicting Fatty Liver Disease. Ann Nutr Metab 2021;77:38-45. doi: 10.1159/000513654.

[2] Ji W, Xue M, Zhang Y, Yao H, Wang Y. A Machine Learning Based Framework to Identify and Classify Non-alcoholic Fatty Liver Disease in a Large-Scale Population. Front Public Health. 2022 Apr 4;10:846118. doi: 10.3389/fpubh.2022.846118. PMID: 35444985; PMCID: PMC9013842.

[3] Wu CC, Yeh WC, Hsu WD, Islam MM, Nguyen PAA, Poly TN, Wang YC, Yang HC, Jack Li YC. Prediction of fatty liver disease using machine learning algorithms. Comput Methods Programs Biomed. 2019 Mar;170:23-29. doi: 10.1016/j.cmpb.2018.12.032. Epub 2018 Dec 29. PMID: 30712601.

[4] Han Ma, Cheng-fu Xu, Zhe Shen, Chao-hui Yu, You-ming Li, "Application of Machine Learning Techniques for Clinical Predictive Modeling: A Cross-Sectional Study on Nonalcoholic Fatty Liver Disease in China", BioMed Research International, vol. 2018, Article ID 4304376, 9 pages, 2018. https://doi.org/10.1155/2018/4304376

[5] M. F. Rabbi, S. M. Mahedy Hasan, A. I. Champa, M. AsifZaman and M. K. Hasan, "Prediction of Liver Disorders using Machine Learning Algorithms: A Comparative Study," 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), Dhaka, Bangladesh, 2020, pp. 111-116, doi: 10.1109/ICAICT51780.2020.9333528.

[6] C. Anuradha, D. Swapna, B. Thati, V. N. Sree and S. P. Praveen, "Diagnosing for Liver Disease Prediction in Patients Using Combined Machine Learning Models," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 889-896, doi: 10.1109/ICSSIT53264.2022.9716312.

[7] Islam, Md & Wu, Chieh-Chen & Poly, Tahmina & Nguyen, Phung Anh & Yang, Hsuan-Chia & Li, Yu-Chuan. (2019). Prediction of Fatty Liver Disease using Machine Learning Algorithms. Computer methods and programs in biomedicine. 170.10.1016/j.cmpb.2018.12.032.

[8] Rahman, A. K. M. & Shamrat, F.M. & Tasnim, Zarrin & Roy, Joy & Hossain, Syed. (2019). A Comparative Study On Liver Disease Prediction Using Supervised Machine Learning Algorithms. 8. 419-422.

[9] El-Shafeiy, Engy & El-Desouky, Ali & Elghamrawy, Sally. (2018). Prediction of Liver

Diseases Based on Machine Learning Technique for Big Data. 10.1007/978-3-319-74690-6_36.

[10] A.M. Hall and A.L. Smith. (1999), “Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper”, In Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference, AAAI Press pp. 235-239.

[11] Torkadi, P.P.; Apte, I.C.; Bhute, A.K. Biochemical evaluation of patients of alcoholic liver disease and non-alcoholic liver disease. Indian J. Clin. Biochem. 2014, 29, 79–83.

[12] Robles-Diaz, M.; Garcia-Cortes, M.; Medina-Caliz, I.; Gonzalez-Jimenez, A.; Gonzalez-Grande, R.; Navarro, J.M.; Castiella, A.; Zapata, E.M.; Romero-Gomez, M.; Blanco, S.; et al. The value of serum aspartate aminotransferase and gamma-glutamyl transpetidase as biomarkers in hepatotoxicity. Liver Int. 2015, 35, 2474–2482.

[13] Arieira, C.; Monteiro, S.; Xavier, S.; Dias de Castro, F.; Magalhaes, J.; Moreira, M.J.; Marinho, C.; Cotter, J. Hepatic steatosis and patients with inflammatory bowel disease: When transient elastography makes the difference. Eur. J. Gastroenterol. Hepatol. 2019, 31, 998–1003

[14] M. Ghosh, M. Mohsin Sarker Raihan, M. Raihan, L. Akter, A. Kumar Bairagi et al., "A comparative analysis of machine learning algorithms to predict liver disease," Intelligent Automation & Soft Computing, vol. 30, no.3, pp. 917–928, 2021.

[15] Ravi Kumar R., Babu Reddy M. and Praveen, 2019 “An evaluation of feature selection algorithms in machine learning”, International journal of scientific & technology research, 8(12) PP. 2071–2074.

[16] Dundar, T.T.; Yurtsever, I.; Pehlivanoglu, M.K.; Yildiz, U.; Eker, A.; Demir, M.A.; Mutluer, A.S.; Tektaş, R.; Kazan, M.S.; Kitis, S.; et al. Machine Learning-Based Surgical Planning for Neurosurgery: Artificial Intelligent Approaches to the Cranium. Front. Surg. 2022, 9, 863633.

[17] Sakatani, K.; Oyama, K.; Hu, L.; Warisawa, S. Estimation of Human Cerebral Atrophy Based on Systemic Metabolic Status Using Machine Learning. Front. Neurol. 2022, 13, 869915.

[18] Yen, H.H.; Wu, P.Y.; Chen, M.F.; Lin, W.C.; Tsai, C.L.; Lin, K.P. Current Status and Future Perspective of Artificial Intelligence in the Management of Peptic Ulcer Bleeding: A Review of Recent Literature. J. Clin. Med. 2021, 10, 3527. [Google Scholar] [CrossRef]

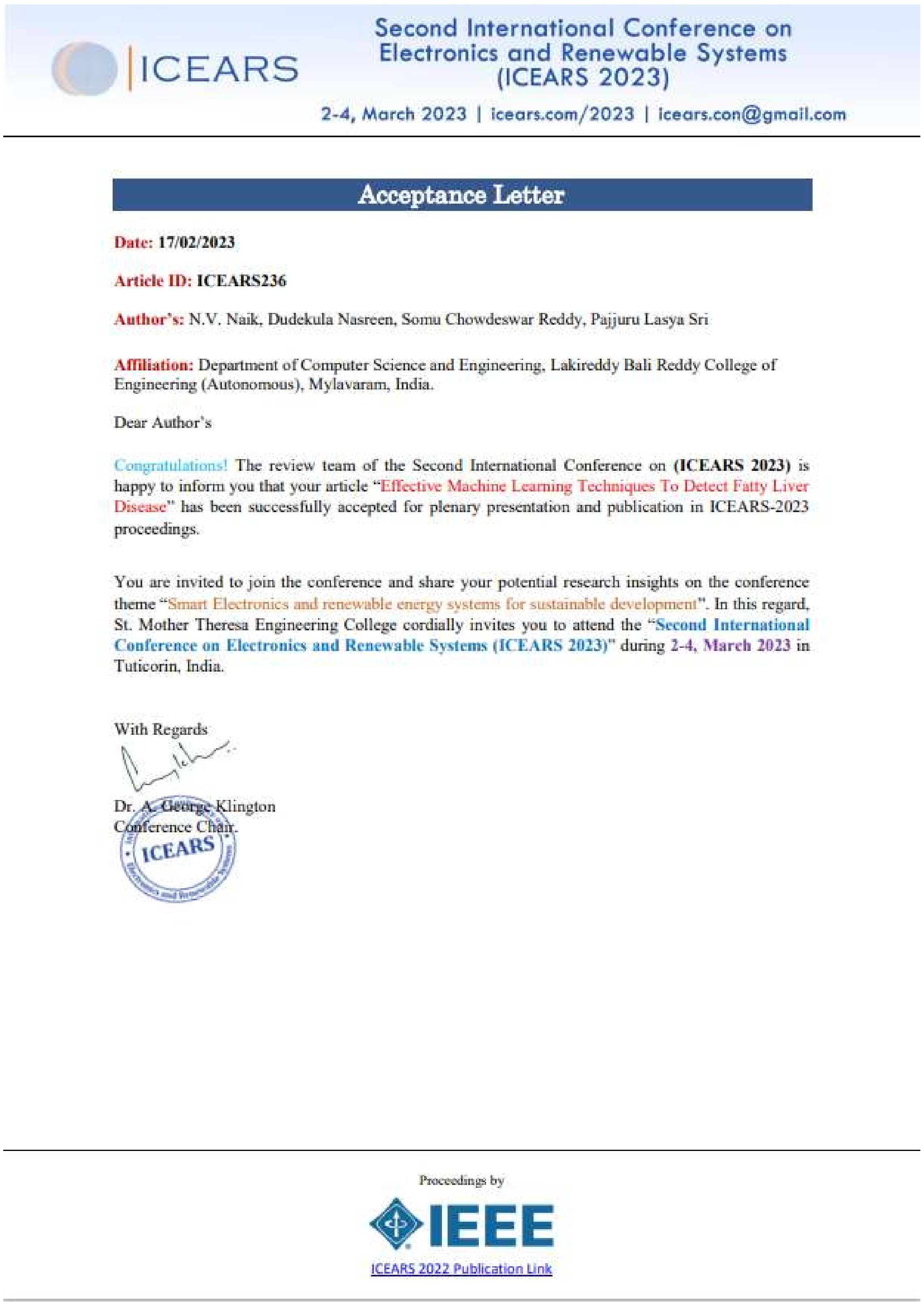
[19] Yen, H.-H.; Wu, P.-Y.; Su, P.-Y.; Yang, C.-W.; Chen, Y.-Y.; Chen, M.-F.; Lin, W.-C.;

Tsai, C.-L.; Lin, K.-P. Performance Comparison of the Deep Learning and the Human Endoscopist for Bleeding Peptic Ulcer Disease. J. Med. Biol. Eng. 2021, 41, 504–513. [Google Scholar] [CrossRef]

[20] Yen, H.H.; Su, P.Y.; Zeng, Y.H.; Liu, I.L.; Huang, S.P.; Hsu, Y.C.; Chen, Y.Y.; Yang, C.W.; Wu, S.S.; Chou, K.C. Glecaprevir-pibrentasvir for chronic hepatitis C: Comparing treatment effect in patients with and without end-stage renal disease in a real-world setting. PLoS ONE 2020, 15, e0237582.

[21] Sakatani, K.; Oyama, K.; Hu, L.; Warisawa, S. Estimation of Human Cerebral Atrophy Based on Systemic Metabolic Status Using Machine Learning. Front. Neurol. 2022, 13, 869915.

[22] Demšar, J.; Curk, T.; Erjavec, A.; Gorup, Č.; Hočevar, T.; Milutinovič, M.; Možina, M.; Polajnar, M.; Toplak, M.; Starič, A.; et al. Orange: Data mining toolbox in Python. J. Mach. Learn. Res. 2013, 14, 2349–2353. [Google Scholar]



ICEARS

Second International Conference on Electronics and Renewable Systems (ICEARS 2023)

2-4, March 2023 | icears.com/2023 | icears.con@gmail.com

Acceptance Letter

Date: 17/02/2023

Article ID: ICEARS236

Author's: N.V. Naik, Dudekula Nasreen, Somu Chowdeswar Reddy, Pajjuru Lasya Sri

Affiliation: Department of Computer Science and Engineering, Lakireddy Bali Reddy College of Engineering (Autonomous), Mylavaram, India.

Dear Author's

Congratulations! The review team of the Second International Conference on **(ICEARS 2023)** is happy to inform you that your article "Effective Machine Learning Techniques To Detect Fatty Liver Disease" has been successfully accepted for plenary presentation and publication in ICEARS-2023 proceedings.

You are invited to join the conference and share your potential research insights on the conference theme "Smart Electronics and renewable energy systems for sustainable development". In this regard, St. Mother Theresa Engineering College cordially invites you to attend the "**Second International Conference on Electronics and Renewable Systems (ICEARS 2023)**" during **2-4, March 2023** in Tuticorin, India.

With Regards

[illegible]

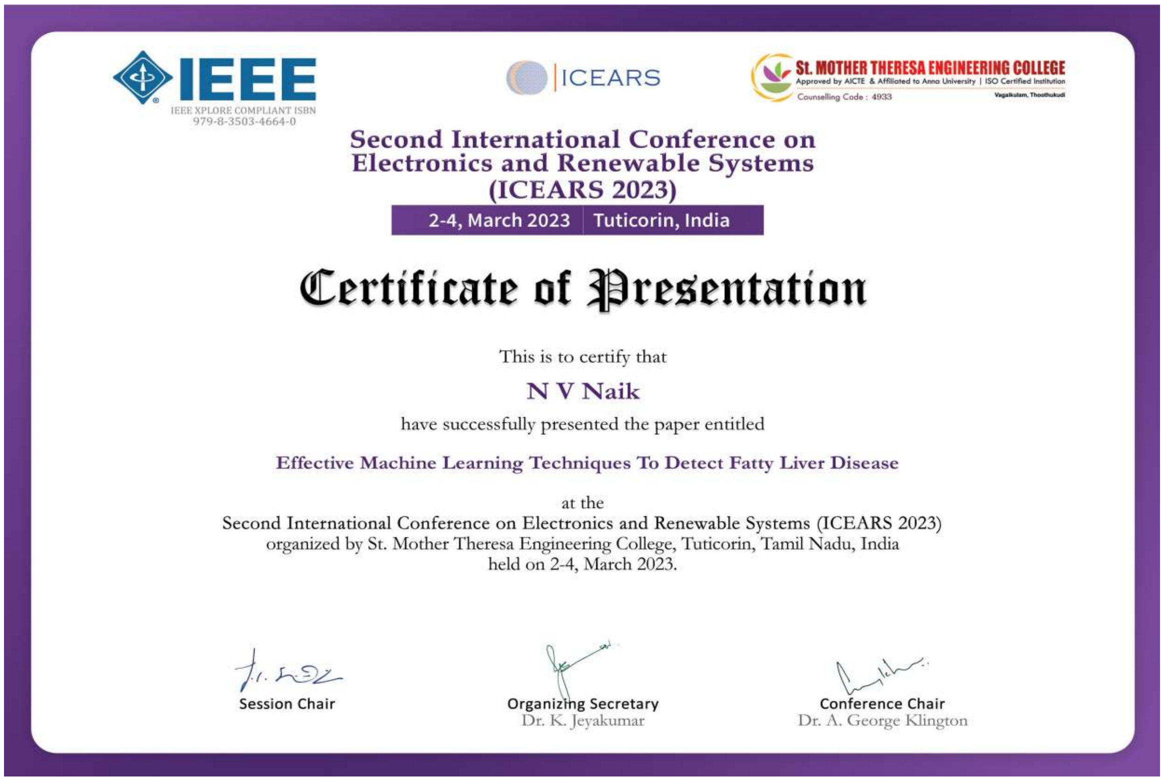
Dr. A. George Klington
Conference Chair.

ICEARS
Electronics and Renewable Systems

Proceedings by

IEEE

ICEARS 2022 Publication Link



IEEE XPLORE COMPLIANT ISBN
979-8-3503-4664-0

ICEARS

St. MOTHER THERESA ENGINEERING COLLEGE
Approved by AICTE & Affiliated to Anna University | ISO Certified Institution
Counselling Code : 4933
Vagaikulam, Thoothukudi

Second International Conference on Electronics and Renewable Systems (ICEARS 2023)

2-4, March 2023 | Tuticorin, India

Certificate of Presentation

This is to certify that

N V Naik

have successfully presented the paper entitled

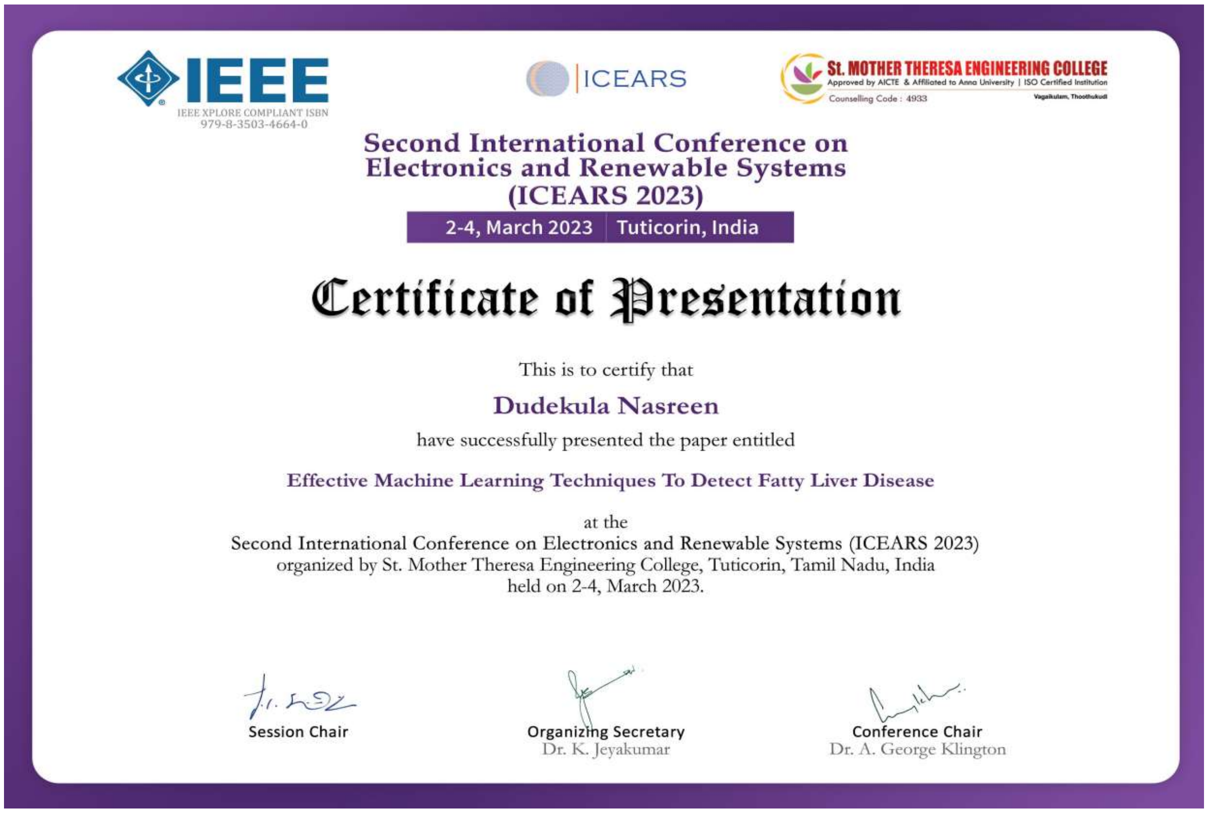
Effective Machine Learning Techniques To Detect Fatty Liver Disease

at the
Second International Conference on Electronics and Renewable Systems (ICEARS 2023)
organized by St. Mother Theresa Engineering College, Tuticorin, Tamil Nadu, India
held on 2-4, March 2023.

[illegible]
Session Chair

[illegible]
Organizing Secretary
Dr. K. Jeyakumar

[illegible]
Conference Chair
Dr. A. George Klington



IEEE XPLORE COMPLIANT ISBN
979-8-3503-4664-0

ICEARS

St. MOTHER THERESA ENGINEERING COLLEGE
Approved by AICTE & Affiliated to Anna University | ISO Certified Institution
Counselling Code : 4933
Vagaikulam, Thoothukudi

Second International Conference on Electronics and Renewable Systems (ICEARS 2023)

2-4, March 2023 | Tuticorin, India

Certificate of Presentation

This is to certify that

Dudekula Nasreen

have successfully presented the paper entitled

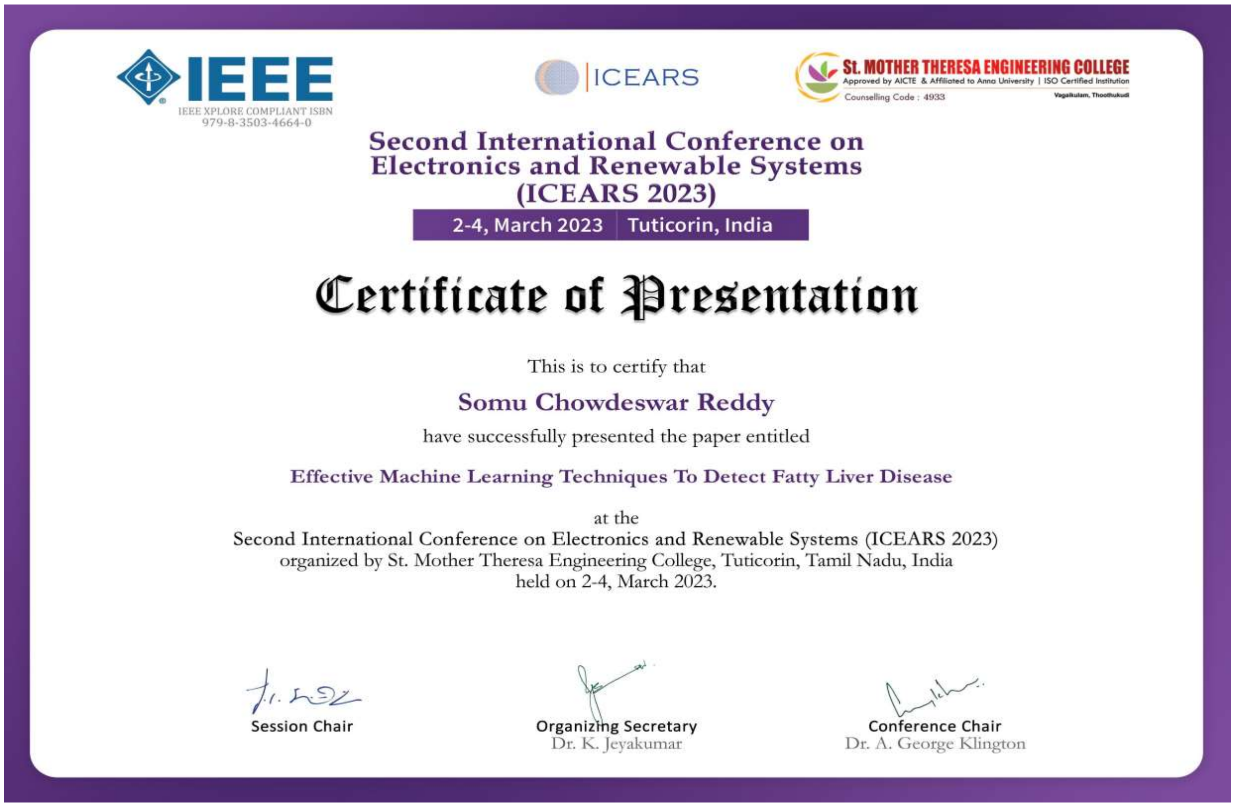
Effective Machine Learning Techniques To Detect Fatty Liver Disease

at the
Second International Conference on Electronics and Renewable Systems (ICEARS 2023)
organized by St. Mother Theresa Engineering College, Tuticorin, Tamil Nadu, India
held on 2-4, March 2023.

[illegible]
Session Chair

[illegible]
Organizing Secretary
Dr. K. Jeyakumar

[illegible]
Conference Chair
Dr. A. George Klington



IEEE XPLORE COMPLIANT ISBN 979-8-3503-4664-0

ICEARS

St. MOTHER THERESA ENGINEERING COLLEGE
Approved by AICTE & Affiliated to Anna University | ISO Certified Institution
Counselling Code : 4933
Vagaikulam, Thoothukudi

Second International Conference on Electronics and Renewable Systems (ICEARS 2023)

2-4, March 2023 | Tuticorin, India

Certificate of Presentation

This is to certify that

Somu Chowdeswar Reddy

have successfully presented the paper entitled

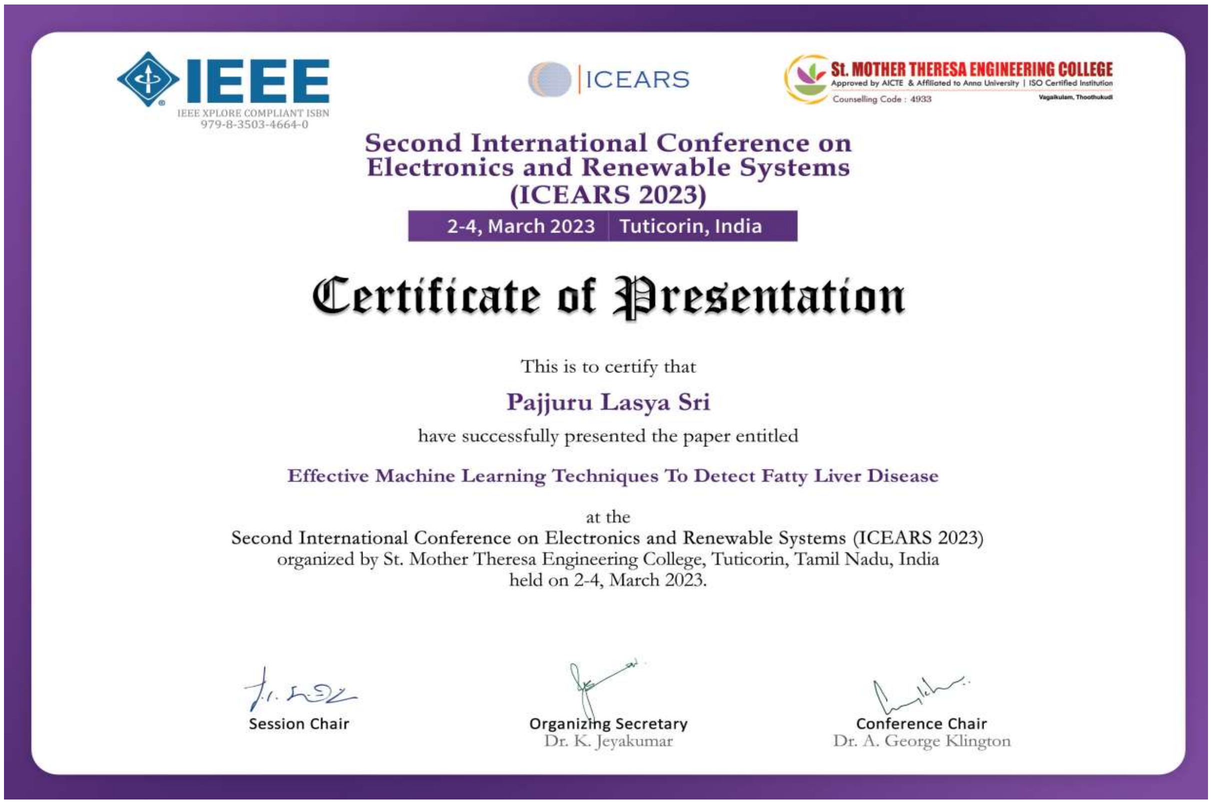
Effective Machine Learning Techniques To Detect Fatty Liver Disease

at the
Second International Conference on Electronics and Renewable Systems (ICEARS 2023)
organized by St. Mother Theresa Engineering College, Tuticorin, Tamil Nadu, India
held on 2-4, March 2023.

[illegible]
Session Chair

[illegible]
Organizing Secretary
Dr. K. Jeyakumar

[illegible]
Conference Chair
Dr. A. George Klington



IEEE XPLORE COMPLIANT ISBN
979-8-3503-4664-0

ICEARS

St. MOTHER THERESA ENGINEERING COLLEGE
Approved by AICTE & Affiliated to Anna University | ISO Certified Institution
Counselling Code : 4933
Vagaikulam, Thoothukudi

Second International Conference on Electronics and Renewable Systems (ICEARS 2023)

2-4, March 2023 | Tuticorin, India

Certificate of Presentation

This is to certify that

Pajjuru Lasya Sri

have successfully presented the paper entitled

Effective Machine Learning Techniques To Detect Fatty Liver Disease

at the
Second International Conference on Electronics and Renewable Systems (ICEARS 2023)
organized by St. Mother Theresa Engineering College, Tuticorin, Tamil Nadu, India
held on 2-4, March 2023.

Session Chair

Organizing Secretary
Dr. K. Jeyakumar

Conference Chair
Dr. A. George Klington